Wire Cell Software Rewrite

Brett Viren

Physics Department

BROOKHAVEN
NATIONAL LABORATORY

BNL Local Meeting 2015 Sep 14

Outline

Packages

Execution Model

Build

Still To Do

Packages

The "productions" packages are now at:

https://github.com/wirecell

- wire-cell-bee, django/JS for Bee event display
- wire-cell-build, top-level C++ build package for:
  - wire-cell-util
  - wire-cell-iface
  - wire-cell-gen
  - wire-cell-alg
  - wire-cell-dfp
  - wire-cell-sst
  - wire-cell-rio
  - wire-cell-riodata
  - wire-cell-rootdict
  - wire-cell-docs

wire-cell-util

- Holds general C++ utility code.
- Depends on no other Wire Cell package.

Some highlights:

Units	system of units
Testing	`Assert()`, memory usage, CPU time
Iterator	abstract iterator
Quantity	simple propagation of uncertainties
3D Vector	object and operators (dot, cross, `*, /, +, −`)
IndexedSet	ordered, unique, random access indexed objects
Configuration	component config via `boost::property_tree`
NamedFactory	plugin instance construction

wire-cell-iface

`iface` = interface:

- Pure abstract base classes.
  - Implementations are not exposed.
- The calling API for Wire Cell.
  - On one side: what LArSoft might code against.
  - On the other: what Wire Cell implementations code against.
- Pervasive use of `shared_ptr<>`.
  - No memory-management hassles.
  - Will have to check performance.

Some highlights

nouns `IData`: wires, cells, blobs, depositions, diffusions, frames, (and eventually slices, tracks, particles.

verbs producers/consumers of `IData`: wire generator, cell maker, framer, digitizer, diffuser, ...

wire-cell-gen

The generator / simulation parts of Wire Cell.

Some highlights:

wires parameter driven wire geometry generation.

cells “bound cell” alg, fast graph-based lookup.

depos depositions: generating, drifting, diffusing.

slices time slices on wires and channels

frames forming frames from slices and vice versa

Work in progress

`wire-cell-*` packages still needing much work:

alg all the actual wire cell reconstruction algorithms. Package exists, but except for hit cells finder (aka `ToyTiling`), empty.

sst celltree geometry and data file reader, in repo, needs porting, needs some interface work.

rio native Wire Cell ROOT input and output, ditto.

... your package here.

Packages

Execution Model

Build

Still To Do

Some questions about “execution model”

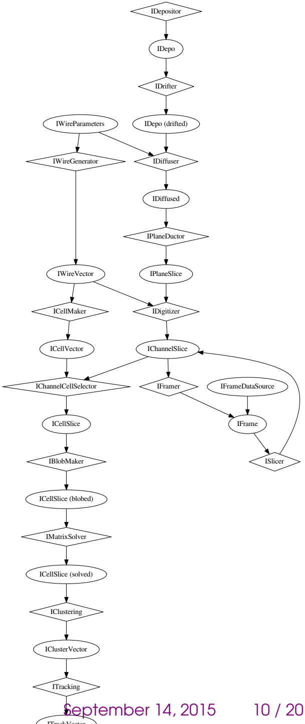
What is an **execution model**?

- How are units of execution defined?
- What drives their execution?
- How are they isolated and how are they coupled?
- Do they run in asynchronously? In parallel?
- How do they access existing data?
- How do they contribute new data?
- How much data exists in memory at once?
- Are there more than one answer to each of the above?

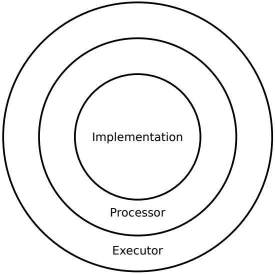
Data Flow Programming (DFP)

The path to fine-grained parallelism while retaining simplicity?

- A graph of **compute vertices** connected by **data edges**.
- Vtx has well defined input/output data types.
- Edges can be thread-safe queues.
- Graph-level programming.
- Can minimize necessary data buffering.
- Graph machinery replaceable: uniproc, multiproc, or distributed (MPI) parallel.
- Encourages isolated, targeted development and testing of each compute vertex.



Execution Model Interface



Each **vertex** in the DFP graph is conceptually a set of concentric circles:

implementation the "guts" of an algorithm with largely unrestricted interface.

processor implementation-specific adapter to outer layer.

executor an execution-model adapter, handles graph definition, drives execution, interface to external execution (ad-hoc, Boost.Pipeline, TBB, MPI, LArSoft).

Common patterns exist at each layer and are exploited to provide simple, reusable base classes for most cases.

Example: single input / single output

```
template<typename InputType, typename OutputType>
class IConverter { public:
  typedef InputType input_type;
  typedef OutputType output_type;

  // Accept an data object for input.
  // Return false if unable to accept.
  virtual bool insert(const input_type& in) = 0;

  // Extract one output data object.
  // Return true if "out" was set successfully.
  virtual bool extract(output_type& out) = 0;
  ...
};
```

- Accepts single input type, produces single output type.
  - Covers most of the required cases.
- Input and output are separate calls: not synchronous!
  - In general, internal buffering needed.

Buffering Protocol (example continued)

```
template<typename InputType, typename OutputType>
class IConverter {
  ...
  // Unconditionally purge all internal buffers.
  virtual void reset() = 0;

  // Flush any remaining input buffers so they are ready for output.
  virtual void flush() = 0;

  // Return an instance of a data object which compares to an
  // end of stream marker.  Implement this if the default
  // output_type instance does not make a suitable EOS marker.
  virtual const output_type& eos() { ... }
};
```

- External `reset()` and `flush()` signals.
  - Allows data stream to be random accessible.
- End-of-stream (EOS) marker sent out after last output object.

Packages

Execution Model

Build

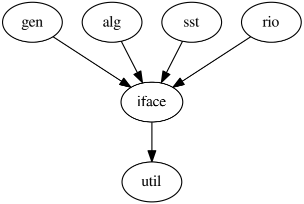
Still To Do

Source/build Changes

- New GitHub org for “production” code: https://github.com/wirecell
- New MkDocs site: http://wirecell.github.io/wire-cell-docs/
- New tool: `wcb` (= “Wire Cell Build”)
  - `wcb` = `waf` + `waf-tools`
  - delete `waf-tools` package
  - same usage: `wcb [...] configure build install`
- Otherwise, everything is the same.

src/	library code
inc/	public headers for library
tests/	**write them as you develop!!!**
apps/	I want to limit the number of apps to ∼1.

Dependency Guidelines



I want to preserve these rules:

1. For “core” packages: `util, iface, gen, alg`
   - only “big” external dependency is **BOOST**.
   - No **ROOT** usage in library code, but OK for tests.
   - Libs for `sst, rio` obviously must depend on ROOT
   - Need ROOT-free: FFT, minimization, and ???
2. Implementation packages do **NOT** depend on each other!
   - Library must only depend on `iface`!
   - May use `NamedFactory` mechanism (think: “plugin”) to find needed implementations (another talk).
   - Again, tests may directly depend on other package.

Packages

Execution Model

Build

Still To Do

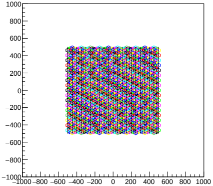
Going forward

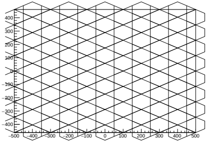
- `wire-cell-gen` almost done
  - Vehicle for solidifying high level concepts of data, interface and execution model.
- Outside contribution possible soon.
- `wire-cell-alg` requires Xin's expert effort
  - Will need "training" to understand new structure. No breaking stuff allowed this time! :)
- Near term: simple uniproc, execution model
- Multiproc using Boost.Pipeline (simple, but experimental package)
- Multiproc using Intel TBB (complex, but established package)
  - Provides really cool/powerful Flow Graph Designer for application visualization and performance tuning. `https://www.youtube.com/watch?v=K4BFpW1NAwo`

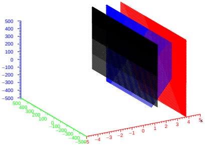
Other stuff

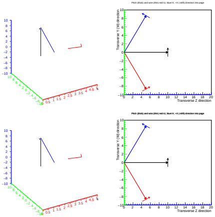
In no particular order:

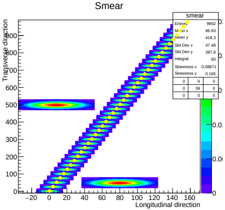
- Need to update documentation.
  - The first “draft” wasn’t much more than a long README and not very well organized.
  - Doxygen build/deployment need automation `http://www.phy.bnl.gov/wire-cell/doxy/html/`
- Understand TBB/Flow Graph Designer.
- Work with LArSoft people to get our requirements met.
- Eventually closeout “prototype” code.
- I’ll present Wire Cell software to BNL CSC on Sept 29th.

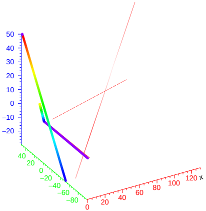

1000
800
600
400
200
0
−200
−400
−600
−800
−1000
−1000 −800 −600 −400 −200 0 200 400 600 800 1000

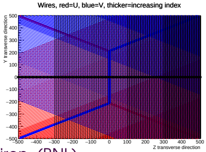

400
300
200
100
0
−100
−200
−300
−400
−500 −400 −300 −200 −100 0 100 200 300 400 500

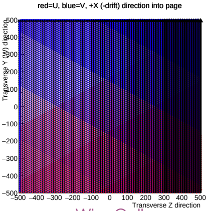

x

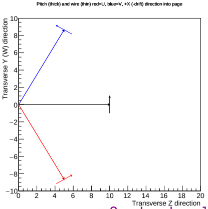

Pitch (thick) and wire (thin) red=U, blue=V, +X (-drift) direction into page
Transverse Y (W) direction
Transverse Z direction


Smear
smear
Entries 9992
Mean x 66.93
Mean y 418.3
Std Dev x 47.48
Std Dev y 287.6
Integral 60
Skewness x 0.08671
Skewness y 0.165
Transverse direction
Longitudinal direction


x


Wires, red=U, blue=V, thicker=increasing index
Y transverse direction
Z transverse direction


red=U, blue=V, +X (-drift) direction into page
Transverse Y (W) direction
Transverse Z direction


Pitch (thick) and wire (thin) red=U, blue=V, +X (-drift) direction into page
Transverse Y (W) direction
Transverse Z direction